

Algoritmos y Estructuras de Datos II

Tercer parcial

Sábado 27 de noviembre de 2021

Aclaraciones

- El parcial es individual en todas sus etapas, y a libro abierto.
- Se responderán consultas por Discord, sólo de interpretación del enunciado, a través del canal ‘Mesa de Docentes’ de la categoría PARCIALES.
- Cualquier aclaración sobre el enunciado se realizará a través del canal ‘anuncios-evaluaciones’.
- El parcial durará 4 horas, de 9 a 13hs, y la entrega se realizará a través del Campus hasta las 13:15hs, en forma estricta. Se desestimarán entregas fuera de tiempo, sin excepciones.
- El ejercicio se calificará con **Perfecto**, **Aprobado** (que puede ir con ‘–’ o ‘?’), **Regular**, o **Insuficiente**.
- Para aprobar la cursada se deben tener todos los ejercicios aprobados de todos los parciales.
- Los ejercicios con **R** o **I** se recuperarán por separado al final del cuatrimestre.
- Sólo se podrá tener un ejercicio con ‘A?’ en la cursada, si no, deberán recuperarse los que hagan falta.

Ej. 1. Sorting

Se está organizando la clásica carrera de San Silvestre y, debido a la situación actual por el coronavirus, se armaron varias tandas de salidas de los corredores. Así, se necesita armar los resultados finales de toda la competencia.

El tiempo de carrera de cada participante se calcula como la diferencia en minutos entre su llegada y su salida, y por cada tanda de corredores se obtendrá una lista ordenada de los participantes según su tiempo de carrera total.

Se desea crear una función que obtenga la lista ordenada de los primeros p corredores más rápidos, a partir de los resultados de todas las tandas:

$$\text{PRIMEROS}(\text{in } p: \text{nat}, \text{ in } \text{tandas}: \text{lista}(\text{lista}(\text{participante}))) \rightarrow \text{res}: \text{lista}(\text{participante})$$

Por ejemplo,

$$\text{PRIMEROS}(3, [[(\text{Ana}, 15), (\text{Beto}, 20)], [(\text{Carlos}, 11), (\text{David}, 12), (\text{Erica}, 30)]]) \rightsquigarrow [(\text{Carlos}, 11), (\text{David}, 12), (\text{Ana}, 15)].$$

- a) Dar un algoritmo que resuelva el problema y cuya complejidad temporal sea $O(t + (p \log t))$, siendo t la cantidad de tandas. Justificar detalladamente que el algoritmo efectivamente resuelve el problema y demostrar formalmente que cumple con la complejidad solicitada.
- b) Explicar detalladamente qué cambiaría en el algoritmo (y la complejidad, sólo de ser estrictamente necesario) si se deseara obtener dos listas con p corredores, una para los profesionales y otra para amateurs suponiendo que este último dato también estaría en la tupla de cada participante (i.e., habrá un booleano que dice si es profesional o no), junto con su nombre y el tiempo de carrera.

Ej. 2. Divide and Conquer

Como parte de un proyecto de generación automática de contenido audiovisual para Internet, se desea que, a partir de una lista dada, combinar aquellos **videos consecutivos** que traten sobre un mismo tema, para obtener películas de la mayor duración posible.

Al momento se cuenta con dos operaciones ya realizadas, las cuales que no se pueden modificar:

- $\text{MISMO_TEMA}(\text{in } v : \text{video}, \text{in } v' : \text{video}) \rightarrow \text{res} : \text{bool}$, que nos dice si dos videos v y v' tratan sobre el mismo tema. Complejidad temporal: $\Theta(1)$.
- $\text{COMBINAR}(\text{in } v : \text{video}, \text{in } v' : \text{video}) \rightarrow \text{res} : \text{video}$, que combina dos videos v y v' . Complejidad temporal: $\Theta(d(v) + d(v'))$, siendo $d(x)$ la duración del video x .

Dar un algoritmo que utilice la técnica *Divide and Conquer* para calcular la función

$$\text{PELICULAS}(\text{in } V : \text{arreglo}(\text{video})) \rightarrow \text{res} : \text{arreglo}(\text{video})$$

Por ejemplo, siendo los videos v_1 , v_3 y v_4 del mismo tema, $\text{PELICULAS}([v_1, v_2, v_3, v_4]) \rightsquigarrow [v_1, v_2, v_{3-4}]$

Se pide que el algoritmo tenga complejidad temporal $O(n + (D \log n))$, donde $D = \sum_{i=1}^n d(V[i])$ (es decir, es la suma de las duraciones de todos los videos). Justificar detalladamente que el algoritmo efectivamente resuelve el problema y demostrar formalmente que cumple con la complejidad solicitada. Para esto último, se deberá utilizar alguno de los métodos vistos de acuerdo a las características del problema que se está resolviendo.